Sri Lanka Institute of Information Technology

Application Frameworks - SE3040

Assignment 01

TimeLabs – Secure RESTful API for a University Timetable Management System

Y3.S2.SE.WE.0201

	Student Registration Number	Student Name
1	IT21318320	Silva T.U.D

Table of Contents

Table of Figures

Introduction

TimeLabs is a fully-fledged and secured RESTful API for a university timetable management application which is built using SpringBoot and for data handling MongoDB has been used. The implemented APIs cover all the aspects of a timetable management system including user management, course management, resource allocation, timetable creation and manipulation, and also notification management. All the RESTful APIs have been tested under different test cases, and also have been tested against security vulnerabilities ensuring the security of the API endpoints. Data manipulation in the APIs has also been secured inside the service functions by ensuring the authorization of users and encrypting sensitive data inside the system to ensure the data privacy at the same time. The system comprises of three types of user roles namely student, faculty, and admin with different levels of authority to access the end points. And the system comprises mainly six components namely User Roles and Authentication, Course Management, Timetable Management, Room and Resource Booking, Student Enrollment, and Notification and Alerts. In summary, the implemented RESTful API ensures the security, scalability, and flexibility in addition to the fully functional of the system which facilitates the users to easily create and use timetable management application.

Assumptions

The APIs was implemented under following assumptions:

- Student will use there registration number as the username when log in to the system.
- Faculty and the admin will use there staff ID as the username when log in to the system.
- Students can be registered to the system without allocating courses to follow and then they can use enrollment API end point to enroll to different courses. (If want they can register to courses at the registration as well)
- Functional requirement is to facilitate weekly timetable creation but in addition to allow creation of timetable for future there have been included a week number to create different timetables for the same batch for different weeks.

User Roles and Authentication

The purpose of the user roles and authentication is to secure the endpoint accessibility to avoid security pitfalls. To ensure the security JWT tokens have been used, and therefore when accessing the endpoints user have to provide the generated token for them. As token consist of details related to user role, it ensures the accessible to endpoints based on the authority level for the given user role.

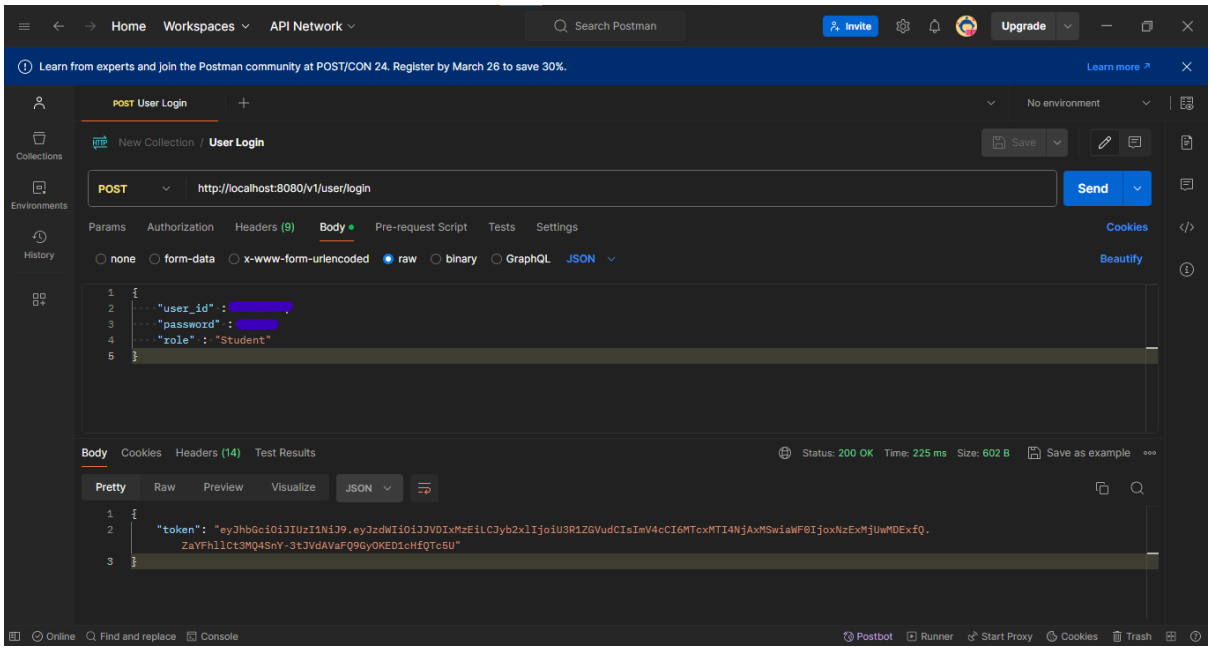


Figure 1: Generation of JWT token

Course Management

Course Management comprises of the following API endpoints to facilitate the creating, updating, deleting, searching for courses. And also to perform those user should have the admin permission.

Endpoint	**Method**
http://localhost:8080/v1/course	POST, GET
http://localhost:8080/v1/course/{code}	GET, PUT, DELETE
http://localhost:8080/v1/{user_id}/enrol	PUT
http://localhost:8080/v1/{user_id}/un-enrol	PUT

Timetable Management

The primary goal of this component is to provide the facility to create timetable slots for a given student batch and then creating the timetable for them by manipulating the data. The following API endpoints facilitates those mentioned functionalities.

Endpoint	**Method**
http://localhost:8080/v1/timetable/slot	POST
http://localhost:8080/v1/timetable/slot/{id}	GET, PUT, DELETE
http://localhost:8080/v1/timetable/slot/{groupId}/{weekNo}	GET
http://localhost:8080/v1/timetable/slot/{groupId}/{weekNo}/{day}	GET
http://localhost:8080/v1/timetable/slot/{lecturer}/{weekNo}	GET
http://localhost:8080/v1/timetable/slot/{lecturer}/{weekNo}/{day}	GET

Room and Resource Booking

The main purpose of this component is to facilitate the rooms and resource booking for a particular timetable slot or for an event without overlapping with already placed booking for the same resources. The following API endpoints facilitates those mentioned functionalities.

Endpoint	**Method**
http://localhost:8080/v1/book-resource	POST
http://localhost:8080/v1/book-resource/{id}	GET, PUT, DELETE
http://localhost:8080/v1/ book-resource	GET

Student Enrollment

This facilitates students to enroll in the courses and view there enrolled courses. And on the other hand, faculty or admin has the capability to view course enrollments, and unenroll them from the courses. The following API endpoints facilitates those mentioned functionalities.

Endpoint	Method
http://localhost:8080/v1/{user-id}/enrol	PUT
http://localhost:8080/v1/{user-id}/un-enrol	PUT
http://localhost:8080/v1/{user-id}	GET
http://localhost:8080/v1/{course-id}	GET

Notifications and Alert

Notifications and alert component controls the sending of notifications to the students and faculties on timetable slot changes. And also, sends notifications regarding the notices. The following API endpoints facilitates those mentioned functionalities.

Endpoint	Method
http://localhost:8080/v1/timetable/slot/{id}	PUT, DELETE
http://localhost:8080/v1/notification/{group-id}	POST

System Overview Diagram

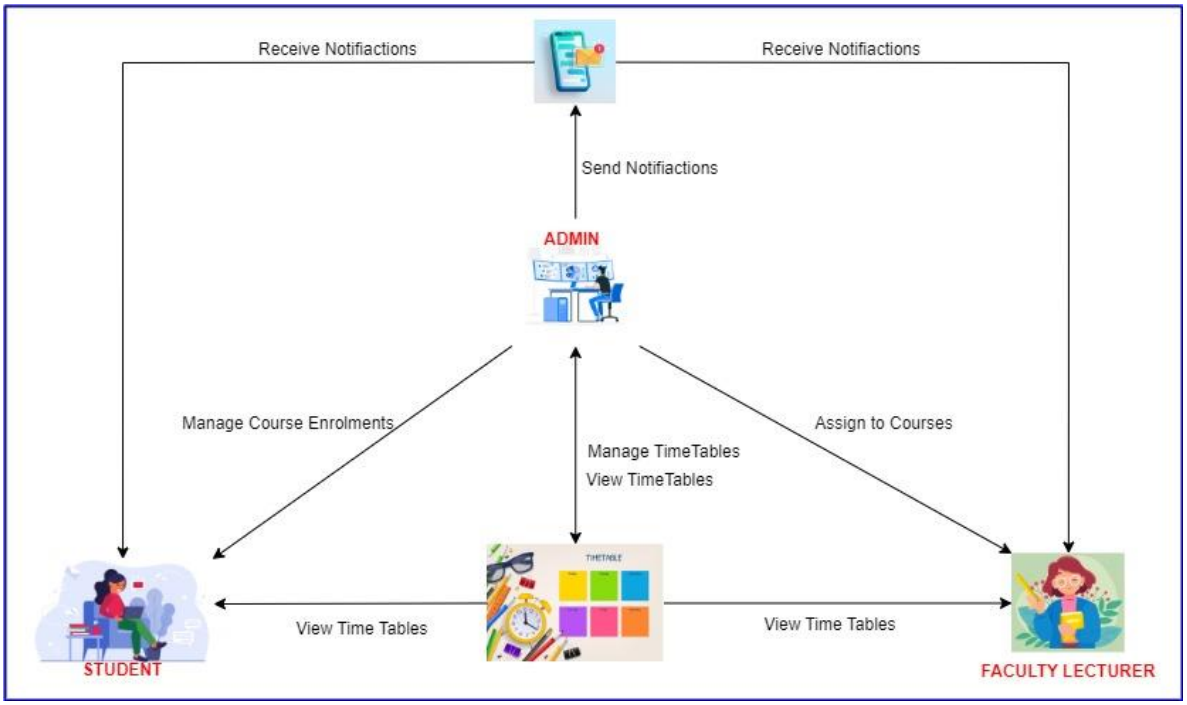


Figure 2: System Overview Diagram

Architecture Diagram

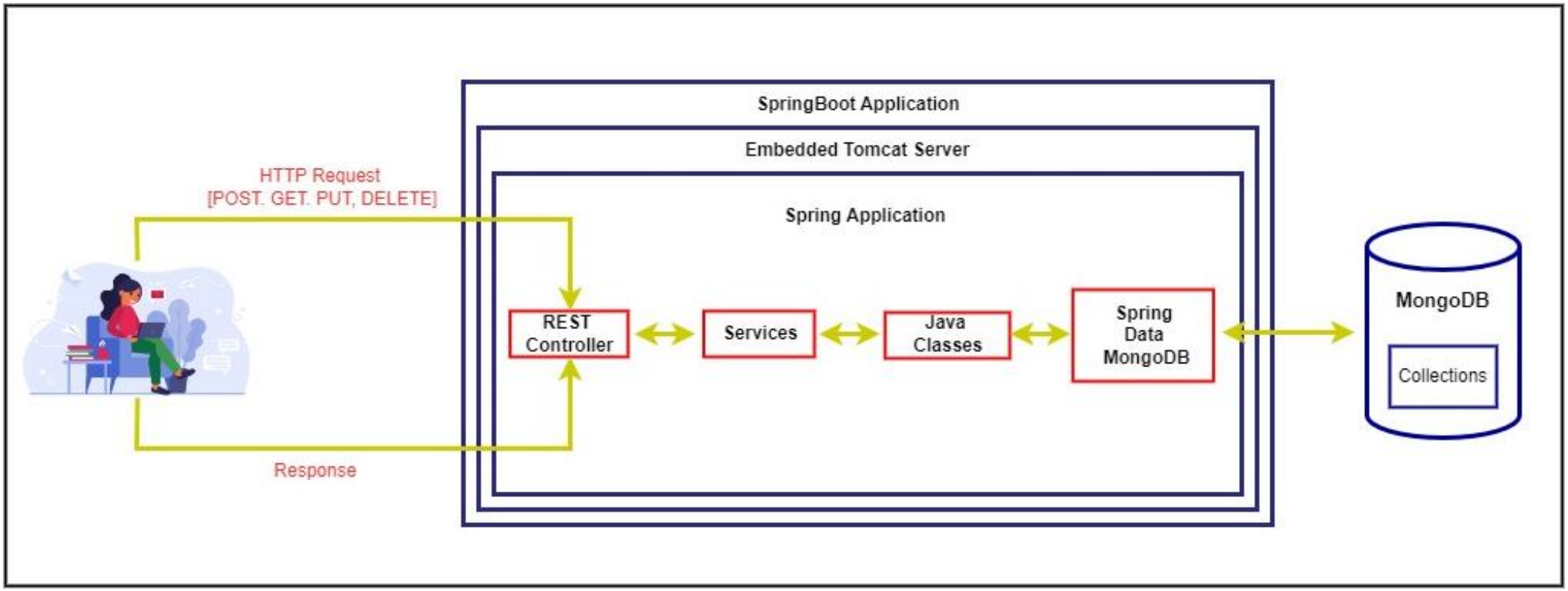


Figure 3: System Architecture Diagram

ER Diagram

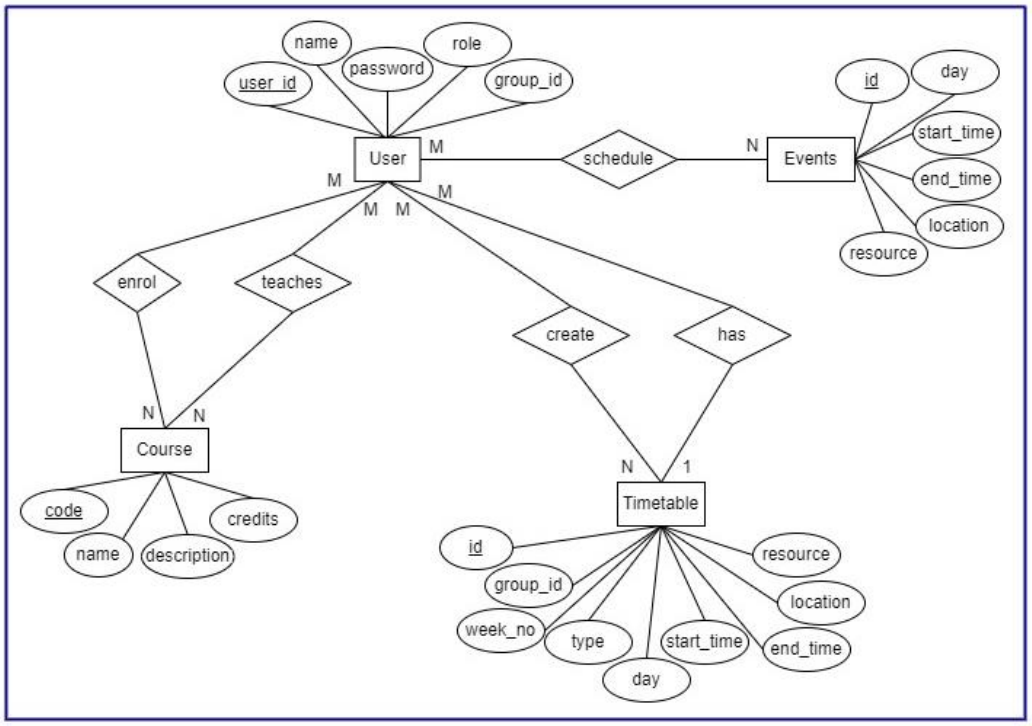


Figure 4: ER Diagram

MongoDB Data Modeling Schema

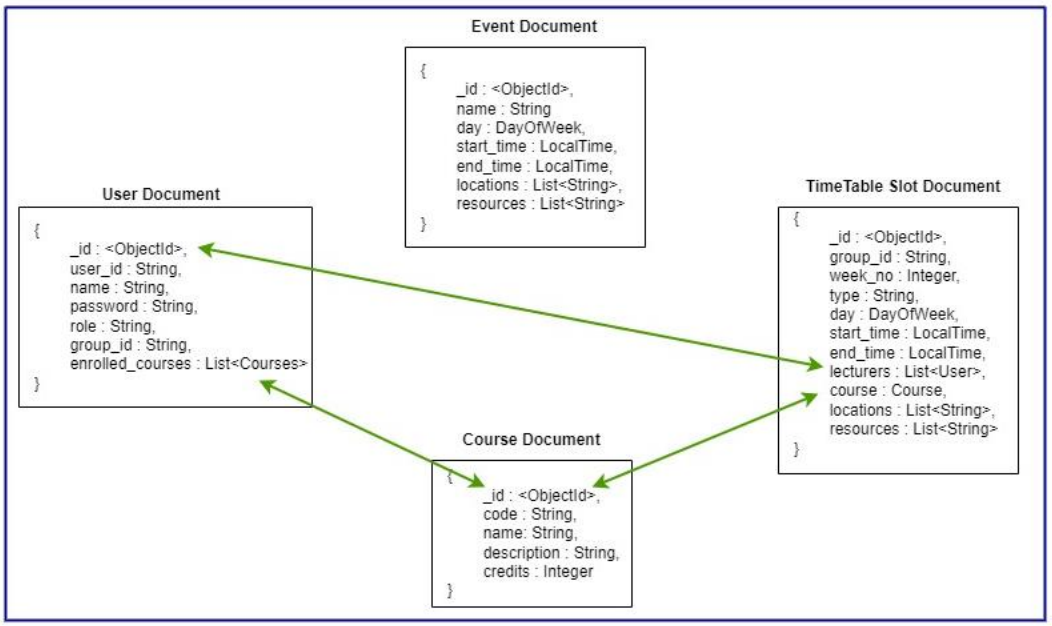
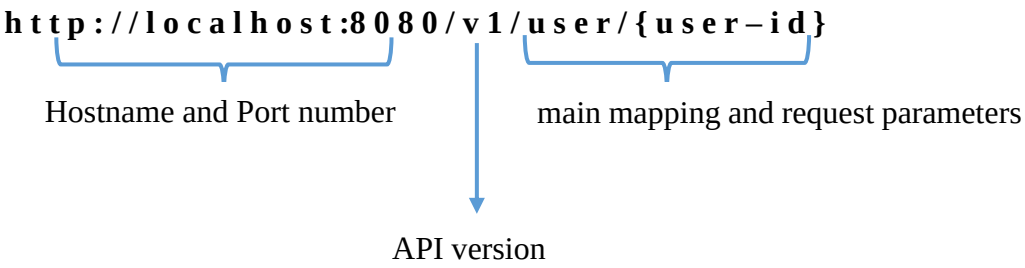


Figure 5: MongoDB Data Modeling Schema

API Endpoint Naming

API endpoint naming has been created based on the following criteria:



API Documentation

API Documentation created for TimeLabs can be found by navigating to the following link.

https://documenter.getpostman.com/view/29491369/2sA35BbjXJ

Folder Structure

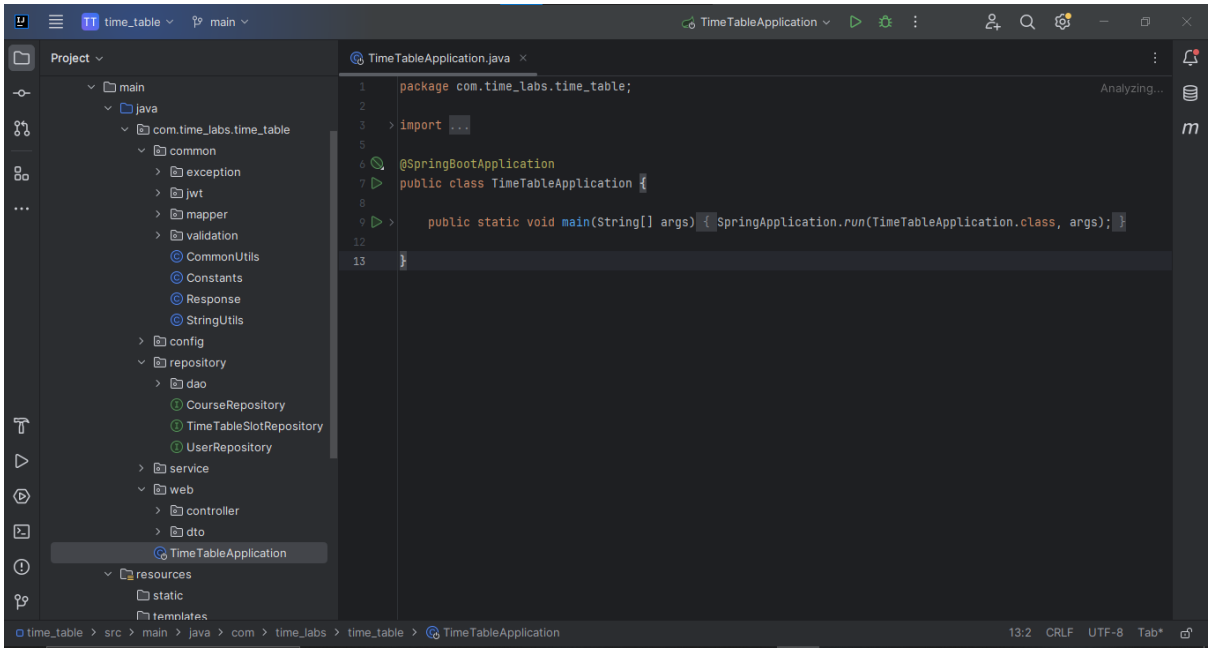


Figure 6: Folder Structure

Unit Testing

Unit test have been created using Junit and Mockito and some sample tests are given below.

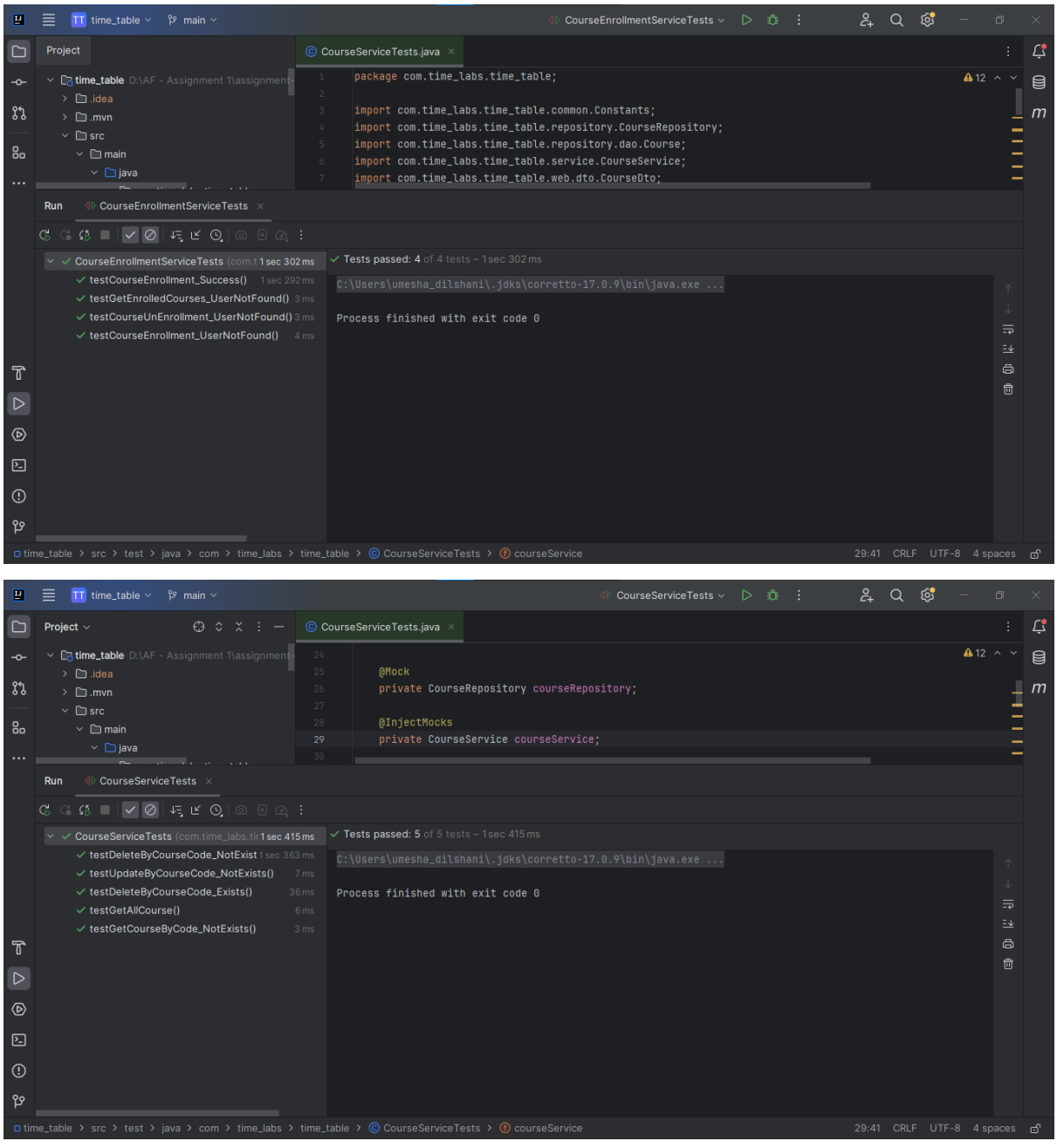
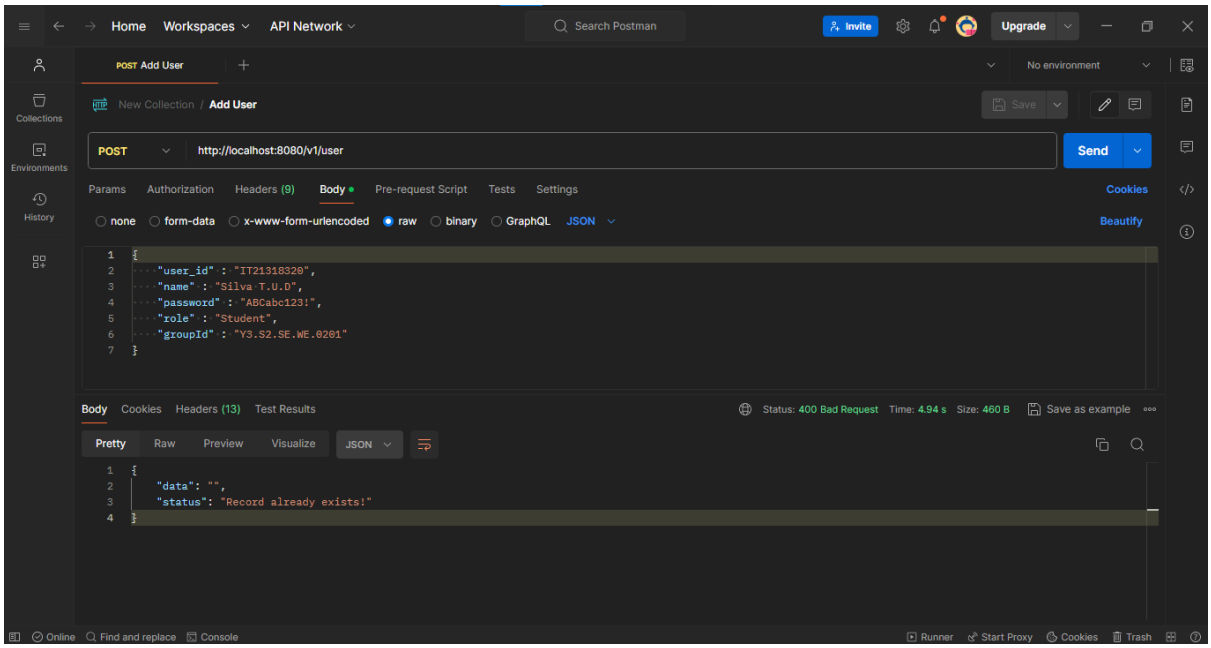


Figure 7: Unit Tests

Integration Testing

Integration testing have been performed using Postman and some sample tests are given below.



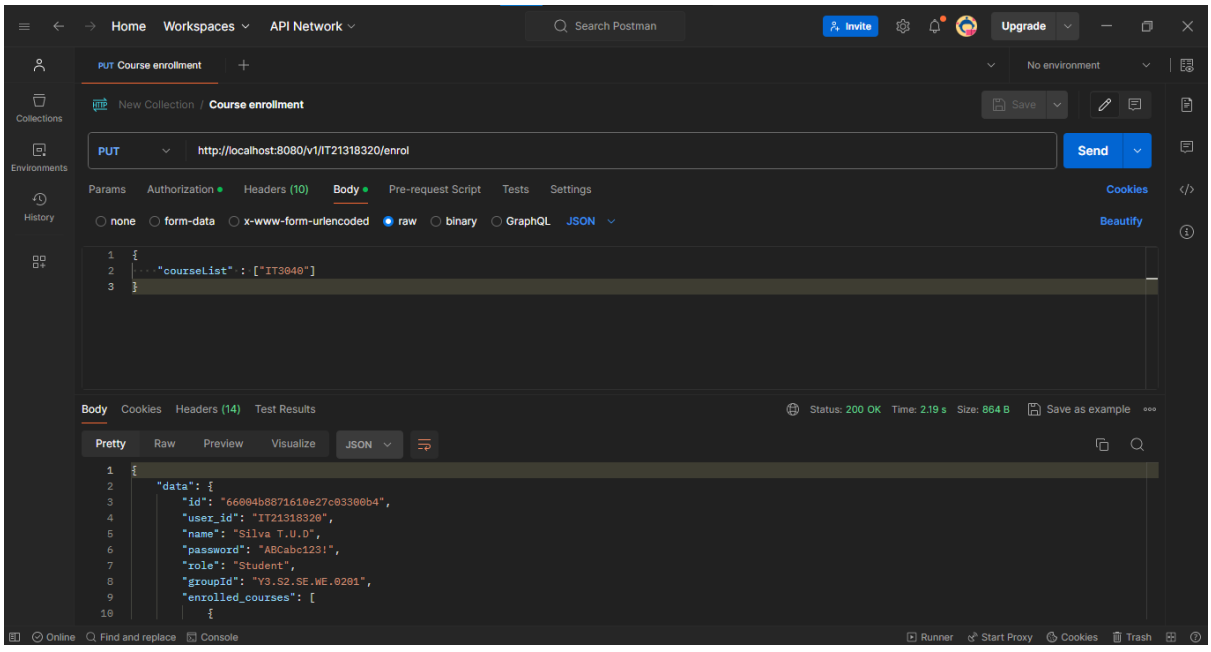


Figure 8: Integration Tests

Security Testing

Security tests have been performed through ZAP and some sample tests are given below.

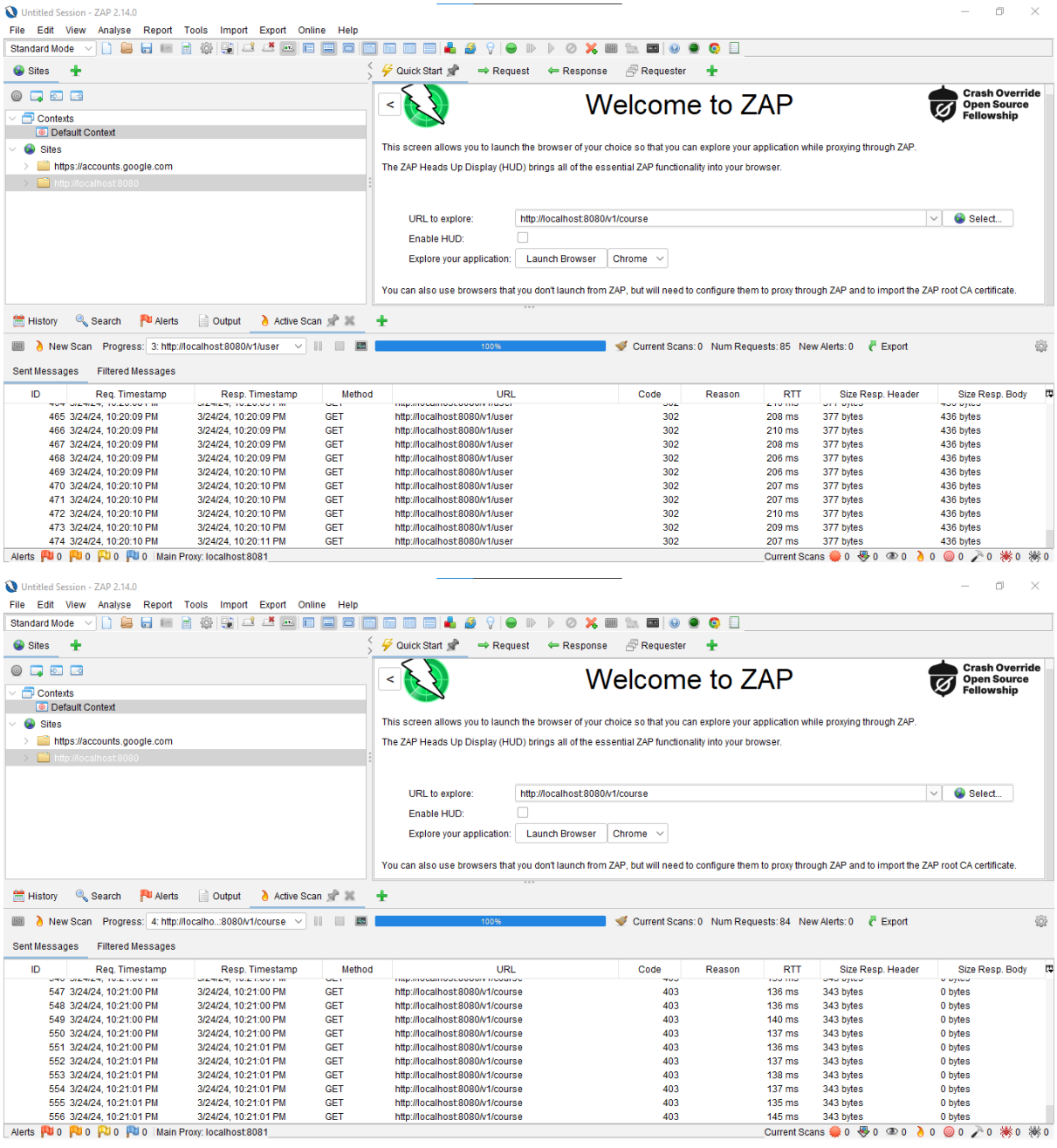


Figure 9: Security Tests